fsmRecord Introduction

State Machines in the Database

Michael Davidsaver

NSLSII Project Brookhaven National Lab

EPICS Collaboration Meeting Fall '10

Outline

Purpose

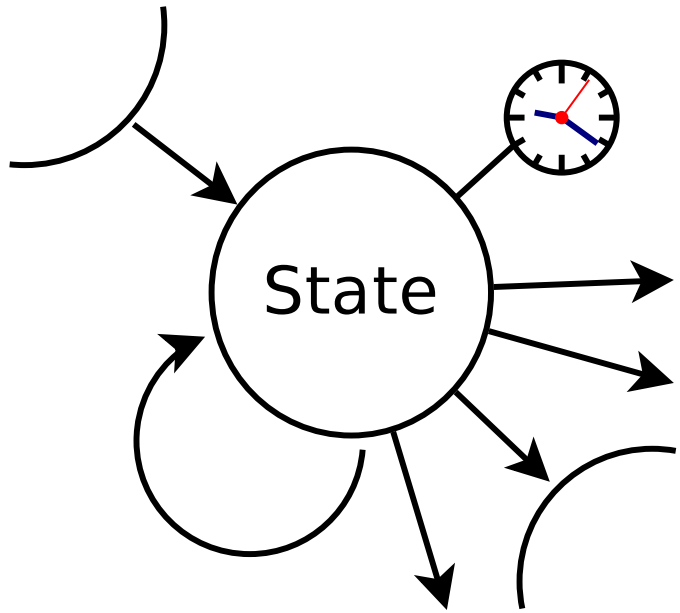
- Represent a finite state machine (FSM) in the EPICS database.
- Allow more runtime reconfiguration
  - Only runtime configuration
- Use standard database techniques and tools

- Source and documentation available

http://sourceforge.net/projects/epics/files/fsmrecord

Finite State Machine

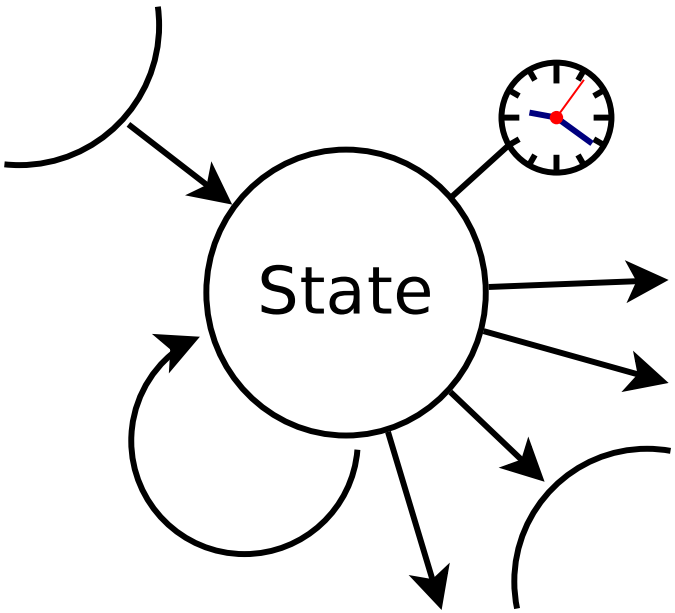
- A FSM is collection of states
- One state is “Active” at a time
- States activated by a transition
  - Token passing
- A state may have several possible transitions



A State Has

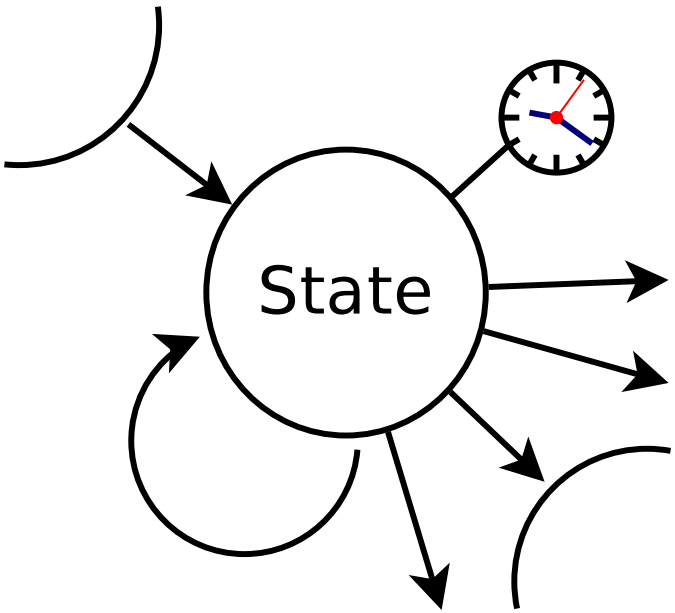
- Transitions
  - Relationship between two states
  - eg. A Kitten can become a Cat
- Conditions
  - Rules to determine when a transition is made
  - eg. A Kitten becomes a Cat when it is fully grown
- Actions
  - Effect on “everything else”, but not the state
  - eg. When a Kitten becomes a Cat it needs a larger collar

Translating into the DB



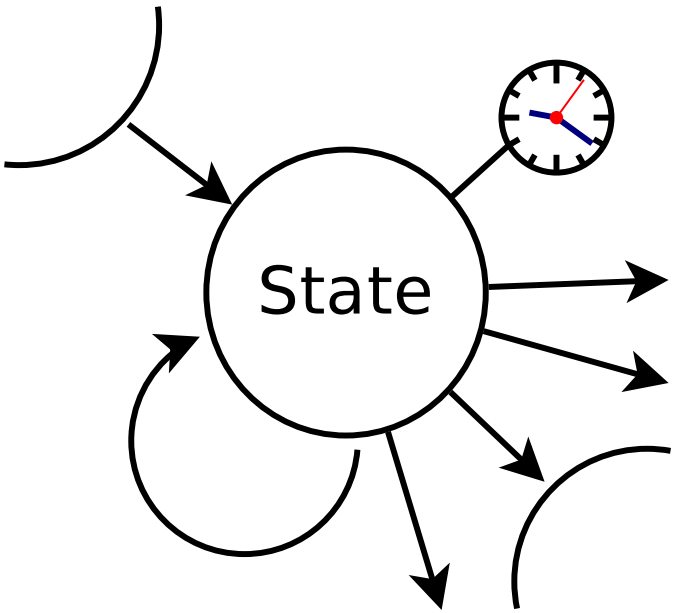
- How to map a FSM into the EPICS database?

Translating into the DB



- How to map a FSM into the EPICS database?
- Each state is a record
- Transitions are links between records

Translating into the DB



- How to map a FSM into the EPICS database?
- Each state is a record
- Transitions are links between records
- Conditions are input links
- Actions are output links

What does it do?

1. Read a series of input links (conditions)
2. Tests the result
3. If false test next condition
4. Otherwise write to the action and transition links

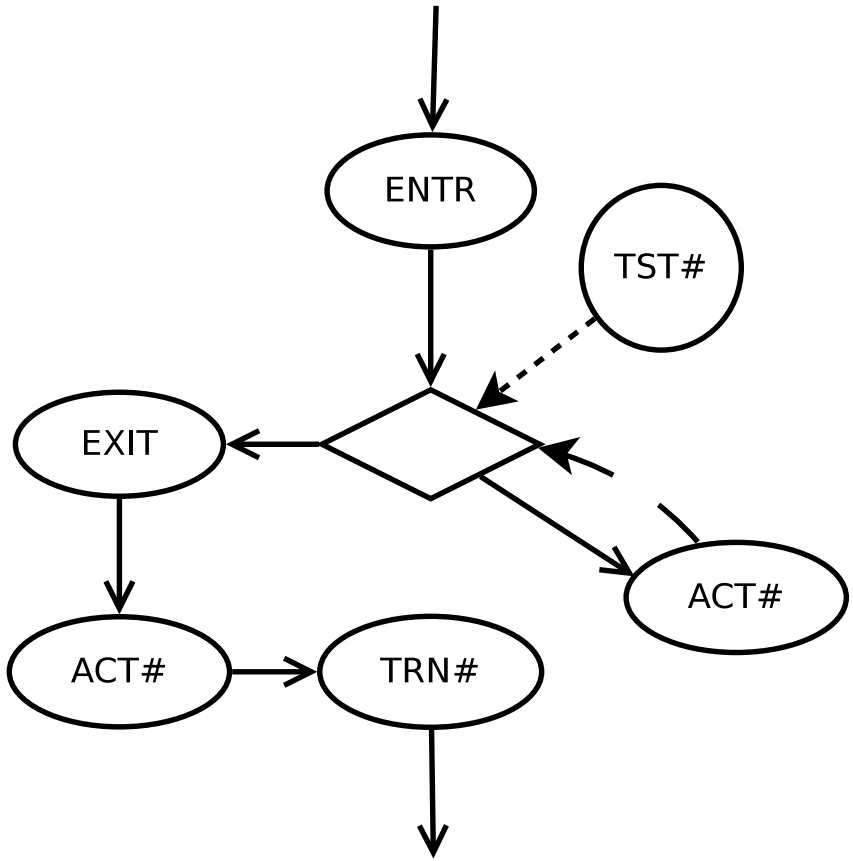
- One record impliments the control logic of one SNL state
- Several records are grouped into a State Set

Another recordtype?

- Why do we need another record type?
  - Always the first question.
- Possible to impliment FSM with only Base record types
  - calc, seq, sel, fanout, ...
- Looks like HDL implimentation
  - Bit mask for actions

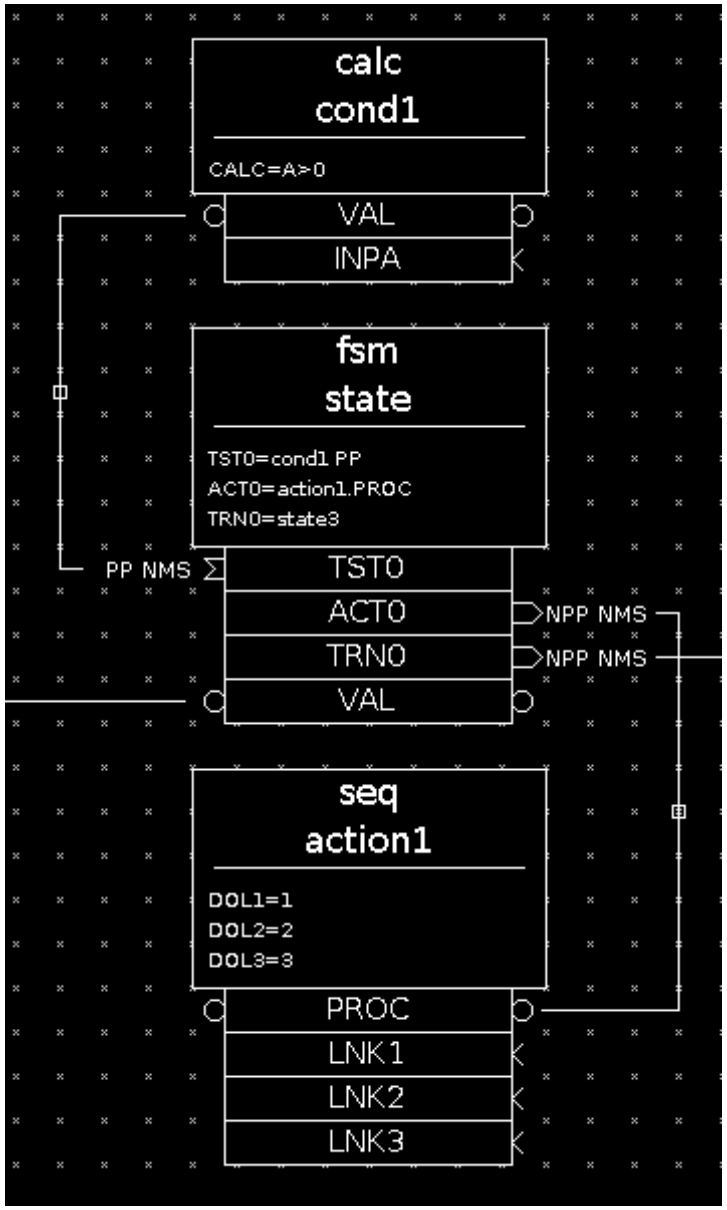
Logic Flow

Condition	
TST#	Test Input link
LVT#	Value read from link
INV#	Invert test logic
Action	
ACT#	Action Output link
AVL#	Value sent to link
ENTR	On enter action
EXIT	On exit action
Transition	
TRN#	Output link



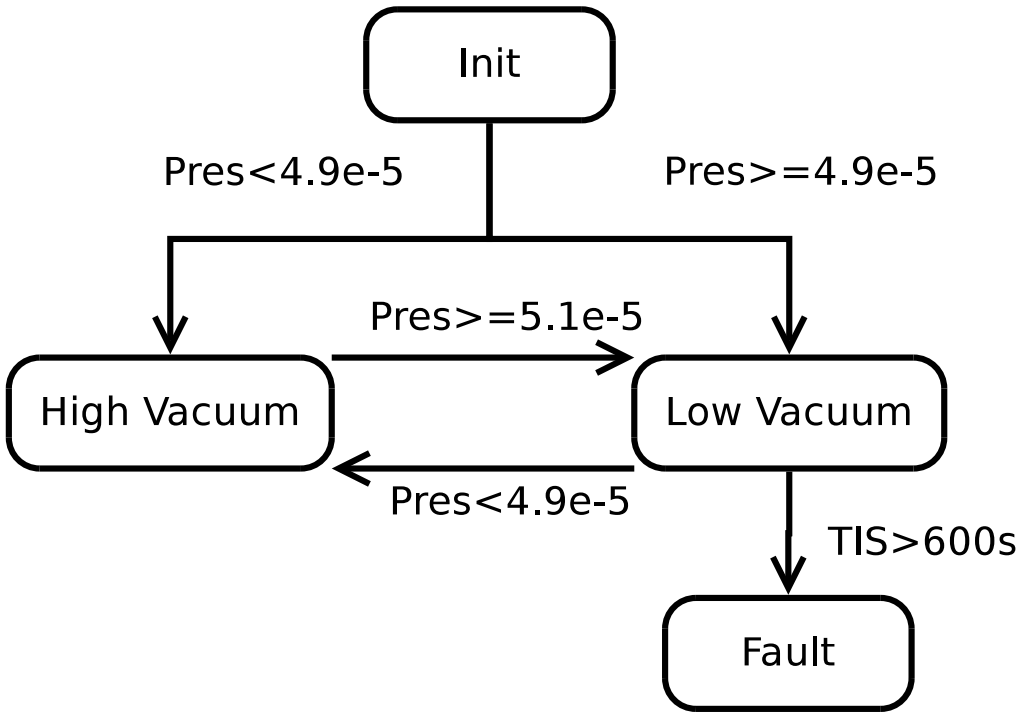
What does it look like?

- Up to 8 transitions
- Each transition has a
  - Test (TST#)
  - Action (ACT#)
  - Transition (TRN#)
- Transitions link to the VAL field of another state
- Tests link to a binary or calc
- seq records for complex actions

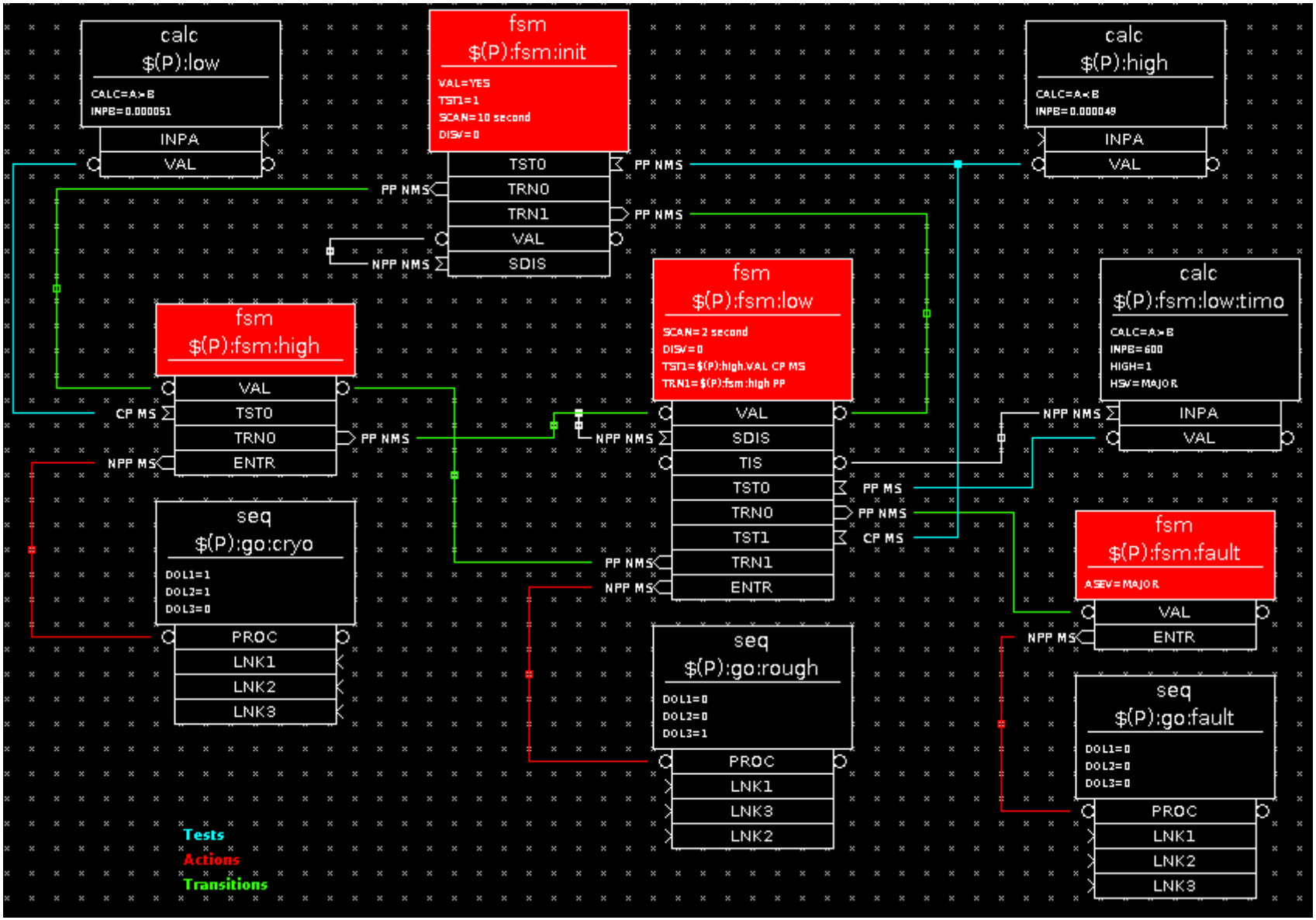


Vacuum pump example (1)

- Example from Ralph Lange
- Devices
  - Rough pump
  - Gate valve
  - Cryogenic pump
- Switch between roughing and cryo pumps



Vacuum pump example (2)



Error Handling

- Errors outside the scope of the FSM logic (bad links)
  - Fail to read new condition value
  - Transition didn't activate another state
- Organize error handling at the State Set level (a collection of states)
- "State Set" is an EPICS driver
  - Transitions must be DB_LINKs
- Failures cause State Set to enter "Reset" mode
  - States deactivated
  - Can't be reactivated
- Controlled via State Set property records

Status and Future Work

- It works, but its new
- Better error handling
- CA link management (wait for connection)

- Where to get it:

http://sourceforge.net/projects/epics/files/fsmrecord

- Thank you